A Proof Search Specification of the π-Calculus

Dale Miller, *INRIA-Futurs and LIX, École Polytechnique*
Alwen Tiu, *LIX and Penn State University*

"A Proof Theory for Generic Judgments" (LICS2003)
"A Proof Search Specification of the π-Calculus" (submitted)

Outline:

1. A proof theoretic approach to term-level binders
2. A meta-logic: sequents, definitions, and ∇-quantification
3. One-step transitions of the π-calculus
4. Open and late simulation

Two slogans

From Alan Perlis's *Epigrams on Programming*: As Will Rogers would have said, "There is no such thing as a free variable."

In our operational semantics specifications: We wish to treat the *names* of binders as the same kind of fiction as we treat *white space*: they are artifacts of how we write expressions and have zero semantic content.

Static and dynamic notions of binders

Modeling the static nature of bindings in expressions seems clear.

● For example, simply typed λ-terms modulo α, β, and η conversions works well to encode syntax. This approach follows the tradition of λ-*tree syntax* (a subset of *higher-order abstract syntax*).

But: how should we model the dynamic nature of binding?

● names in the π-calculus, nonces and keys in security protocols, locations in references, etc.

If computations are modeled using proof search, term-level bindings move to proof-level bindings (mobility of binders).

● Eigenvariables are one form of proof level abstractions. We argue that these are not sufficient.

● A new *local* binding context and the ∇-quantifier will be introduced to provide for another kind of proof-level binder.

The collapse of eigenvariables

An attempt to build a cut-free proof of $\forall x \forall y. P\, x\, y$ first introduces two new and different eigenvariables c and d and then attempts to prove $P\, c\, d$.

Eigenvariables have been used to encode names in π-calculus [Miller93], nonces in security protocols [Cervesato, et.al. 99], reference locations in imperative programming [Chirimar95], etc.

Since $\forall x \forall y. P\ x\ y \supset \forall z. P\ z\ z$ is provable, it follows that the provability of $\forall x \forall y. P\, x\, y$ implies the provability of $\forall z. P\, z\, z$. That is, there is also a cut-free proof where the eigenvariables c and d are identified.

Thus, eigenvariables are unlikely to capture the proper logic behind things like nonces, references, names, etc.

Generic judgments and a new quantifier

Should $\forall x$ really mean a (possibly infinite) conjunction indexed by terms? One reading of Gentzen's introduction rules allows this.

We introduce the quantification $\nabla x. B\, x$ for the more "intensional", "generic", or "internal" reading, and a new local context into sequents.

$$\Sigma : B_1, \ldots, B_n \longrightarrow B_0$$

$$\Downarrow$$

$$\Sigma : \sigma_1 \triangleright B_1, \ldots, \sigma_n \triangleright B_n \longrightarrow \sigma_0 \triangleright B_0$$

Σ is a list of distinct eigenvariables, scoped over the sequent and σ_i is a list of distinct variables, locally scoped over the formula B_i.

The expression $\sigma_i \triangleright B_i$ is called a *generic judgment*. Equality between judgments is defined up to renaming of local variables.

Intuitionistic logic with ∇

$$\frac{}{\Sigma : \mathcal{B} \longrightarrow \mathcal{B}}\ init \qquad \frac{\Sigma : \Delta \longrightarrow \mathcal{B} \quad \Sigma : \mathcal{B}, \Gamma \longrightarrow \mathcal{C}}{\Sigma : \Delta, \Gamma \longrightarrow \mathcal{C}}\ cut$$

$$\frac{\Sigma : \mathcal{B}, \mathcal{B}, \Gamma \longrightarrow \mathcal{C}}{\Sigma : \mathcal{B}, \Gamma \longrightarrow \mathcal{C}}\ c\mathcal{L} \qquad \frac{\Sigma : \Gamma \longrightarrow \mathcal{C}}{\Sigma : \mathcal{B}, \Gamma \longrightarrow \mathcal{C}}\ w\mathcal{L}$$

$$\frac{}{\Sigma : \sigma \triangleright \bot, \Gamma \longrightarrow \mathcal{B}}\ \bot\mathcal{L} \qquad \frac{}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \top}\ \top\mathcal{R}$$

$$\frac{\Sigma : \sigma \triangleright B_i, \Gamma \longrightarrow \mathcal{D}}{\Sigma : \sigma \triangleright B_1 \wedge B_2, \Gamma \longrightarrow \mathcal{D}} \qquad \frac{\Sigma : \Gamma \longrightarrow \sigma \triangleright B_1 \quad \Sigma : \Gamma \longrightarrow \sigma \triangleright B_2}{\Sigma : \Gamma \longrightarrow \sigma \triangleright B_1 \wedge B_2}$$

$$\frac{\Sigma : \sigma \triangleright B_1, \Gamma \longrightarrow \mathcal{D} \quad \Sigma : \sigma \triangleright B_2, \Gamma \longrightarrow \mathcal{D}}{\Sigma : \sigma \triangleright B_1 \vee B_2, \Gamma \longrightarrow \mathcal{D}} \qquad \frac{\Sigma : \Gamma \longrightarrow \sigma \triangleright B_i}{\Sigma : \Gamma \longrightarrow \sigma \triangleright B_1 \vee B_2}$$

$$\frac{\Sigma : \Gamma \longrightarrow \sigma \triangleright B \quad \Sigma : \sigma \triangleright C, \Gamma \longrightarrow \mathcal{D}}{\Sigma : \sigma \triangleright B \supset C, \Gamma \longrightarrow \mathcal{D}} \qquad \frac{\Sigma : \sigma \triangleright B, \Gamma \longrightarrow \sigma \triangleright C}{\Sigma : \Gamma \longrightarrow \sigma \triangleright B \supset C}$$

The quantifiers

$$\frac{\Sigma : (\sigma, y : \tau) \triangleright B[y/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma : \sigma \triangleright \nabla_\tau x.B, \Gamma \longrightarrow \mathcal{C}} \qquad \frac{\Sigma : \Gamma \longrightarrow (\sigma, y : \tau) \triangleright B[y/x]}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \nabla_\tau x.B}$$

$$\frac{\Sigma, \sigma \vdash t : \gamma \quad \Sigma : \sigma \triangleright B[t/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma : \sigma \triangleright \forall_\gamma x.B, \Gamma \longrightarrow \mathcal{C}} \qquad \frac{\Sigma, h : \Gamma \longrightarrow \sigma \triangleright B[(h\ \sigma)/x]}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \forall x.B}$$

$$\frac{\Sigma, h : \sigma \triangleright B[(h\ \sigma)/x], \Gamma \longrightarrow \mathcal{C}}{\Sigma : \sigma \triangleright \exists x.B, \Gamma \longrightarrow \mathcal{C}} \qquad \frac{\Sigma, \sigma \vdash t : \gamma \quad \Sigma : \Gamma \longrightarrow \sigma \triangleright B[t/x]}{\Sigma : \Gamma \longrightarrow \sigma \triangleright \exists_\gamma x.B}$$

Terms are assumed to be simply typed. Dependency between eigenvariables and the local context is encoded using *raising* [Miller92]. For example:

$$\cfrac{\cfrac{\{x_\alpha, h_{\tau\to\gamma\to\beta}\} : \Gamma \longrightarrow (a_\tau, b_\gamma) \triangleright B\ (h\,a\,b)\ b}{\{x_\alpha\} : \Gamma \longrightarrow (a_\tau, b_\gamma) \triangleright \forall_\beta y.B\ y\ b}\ \forall\mathcal{L}}{\{x_\alpha\} : \Gamma \longrightarrow (a_\tau) \triangleright \nabla_\gamma z.\forall_\beta y.B\ y\ z}\ \nabla\mathcal{R}$$

Properties of ∇

Some theorems:

$$\nabla x \neg Bx \equiv \neg \nabla x Bx \qquad \nabla x(Bx \wedge Cx) \equiv \nabla x Bx \wedge \nabla x Cx$$

$$\nabla x(Bx \vee Cx) \equiv \nabla x Bx \vee \nabla x Cx \qquad \nabla x(Bx \supset Cx) \equiv \nabla x Bx \supset \nabla x Cx$$

$$\nabla x \forall y Bxy \equiv \forall h \nabla x Bx(hx) \qquad \nabla x \exists y Bxy \equiv \exists h \nabla x Bx(hx)$$

$$\nabla x \forall y Bxy \supset \forall y \nabla x Bxy \qquad \nabla x.\top \equiv \top, \quad \nabla x.\bot \equiv \bot$$

Consequence: ∇ can always be given atomic scope within formulas.

Some non-theorems:

$$\nabla x \nabla y Bxy \supset \nabla z Bzz \qquad \nabla x Bx \supset \exists x Bx$$

$$\nabla z Bzz \supset \nabla x \nabla y Bxy \qquad \forall x Bx \supset \nabla x Bx$$

$$\forall y \nabla x Bxy \supset \nabla x \forall y Bxy \qquad \exists x Bx \supset \nabla x Bx$$

$$\nabla x Bx \supset \forall x Bx \qquad \nabla x B \equiv B \qquad \nabla x \nabla y.B\,x\,y \equiv \nabla y \nabla x.B\,x\,y$$

A proof theoretic notion of definitions

We extend the logic further by allowing non-logical constants (predicate) to be introduced. To each predicate, we associate some collection of *definition clauses*. We write

$$\forall \bar{x}. p\,\bar{t} \triangleq B$$

to denote a definition clause for predicate p. Free variables in B are in the set of free variables in $\bar{t}$, which are all in $\bar{x}$. This notion of definition has been previously studied in proof theory by Schroeder-Heister, Girard (fixpoints), McDowell/Miller/Tiu.

Closely related to the "Clark completion" studied in logic programming.

By imposing certain restriction on definitions, we can prove cut-elimination.

Introduction rules for definitions

Without ∇, the right introduction rule for an atomic formula A is

$$\frac{\Sigma : \Gamma \longrightarrow B\theta}{\Sigma : \Gamma \longrightarrow A}\ \mathit{def}\mathcal{R}$$

provided that there is a clause $\forall \bar{x}.[H \stackrel{\triangle}{=} B]$ such that $A = (H\theta)$.

The left introduction rule ($\mathit{def}\mathcal{L}$) is

$$\frac{\{\Sigma\theta : B\theta, \Gamma\theta \longrightarrow C\theta \mid \forall \bar{x}.[H \stackrel{\triangle}{=} B] \text{ is a def clause, } \theta \in csu(A, H)\}}{\Sigma : A, \Gamma \longrightarrow C}$$

The set of premise set can be empty. *csu* stands for "complete set of unifiers."

The signature $\Sigma\theta$ is obtained from Σ by removing variables in the domain of θ, and adding free variables in the range of θ.

Notice that *eigenvariables can be instantiated*.

Examples: $1+2=3$ and $1+2\neq 1$

Define addition on a simple encoding of natural numbers.

$$\begin{array}{rcl} sum\ z\ N\ N & \triangleq & \top. \\ sum\ (s\ N)\ M\ (s\ P) & \triangleq & sum\ N\ M\ P. \end{array}$$

We can prove that $1+2=3$

$$\dfrac{\dfrac{\longrightarrow \top}{\longrightarrow sum\ z\ (s\ (s\ z))\ (s\ (s\ z))}\ def\mathcal{R}}{\longrightarrow sum\ (s\ z)\ (s\ (s\ z))\ (s\ (s\ (s\ z)))}\ def\mathcal{R}$$

and that $1+2\neq 1$.

$$\dfrac{\dfrac{}{sum\ z\ (s\ (s\ z))\ z \longrightarrow}\ def\mathcal{L}}{sum\ (s\ z)\ (s\ (s\ z))\ (s\ z) \longrightarrow}\ def\mathcal{L}$$

More generally, consider (where n and m are eigen-variables):

$$\frac{\Gamma(n,n), \top \longrightarrow G(n,n)}{\Gamma(n,m), sum\ z\ n\ m \longrightarrow G(n,m)}\ def\mathcal{L}$$

Example: computing max

Let the predicate a be given by a table, for example, as:

$$\begin{aligned} a\ (s\ z) &\triangleq \top. \\ a\ (s\ (s\ (s\ z))) &\triangleq \top. \\ a\ z &\triangleq \top. \end{aligned}$$

Specify $maxa\ N$ so that this is provable if and only if N is the maximum argument for a.

This is impossible if we use logic programs (instead of definitions) because of monotonicity. Definitions with $def\mathcal{L}$ provides a solution.

This specification will work. And it's the natural one!

$$\begin{aligned} maxa\ N &\triangleq (a\ N) \wedge \forall x(a\ x \supset x \leq N). \\ z \leq N &\triangleq true. \\ (s\ N) \leq (s\ M) &\triangleq N \leq M. \end{aligned}$$

Applying definitions to judgments

Definitions can be “raised” so that they are defining generic judgments and not just atomic formulas.

Given a definition clause $\forall \bar{x}.H \triangleq B$, and a list of variables $\bar{y}$, its raised form w.r.t. $\bar{y}$ is (this is similar to $\forall$-lifting in Isabelle):

$$\forall \bar{h}.\ \bar{y} \triangleright H[(\bar{h}\,\bar{y})/\bar{x}] \triangleq \bar{y} \triangleright B[(\bar{h}\,\bar{y})/\bar{x}].$$

The right introduction rule for a judgment $\bar{y} \triangleright A$

$$\frac{\Sigma : \Gamma \longrightarrow (\bar{y} \triangleright B)\theta}{\Sigma : \Gamma \longrightarrow \bar{y} \triangleright A}\ \mathit{def}\mathcal{R}$$

where $\forall \bar{h}.\ \bar{y} \triangleright H \triangleq \bar{y} \triangleright B$ is a raised definition clause and

$$\lambda \bar{y}.A =_{\beta\eta} (\lambda \bar{y}.H)\theta.$$

The left rule is given by

$$\frac{\{\Sigma\theta : (\bar{y} \triangleright B)\theta, \Gamma\theta \longrightarrow \mathcal{C}\theta\}_{B,\theta}}{\Sigma : \bar{y} \triangleright A, \Gamma \longrightarrow \mathcal{C}}\ \mathit{def}\mathcal{L}$$

where $\forall \bar{h}.\bar{y} \triangleright H \triangleq \bar{y} \triangleright B$ is a raised definition clause and

$$(\lambda\bar{y}.A)\theta =_{\beta\eta} (\lambda\bar{y}.H)\theta.$$

Notice that *the local variables $\bar{y}$ are not instantiated*.

No new "technology" is needed here. We rely on the unification of simply typed λ-terms. Such unification is often "easy" (meaning, L_λ unification or *higher-order patterns unification*). This is the case, for example, with the π-calculus examples.

Meta theorems

Theorem: *Cut-elimination.* Given a fixed stratified definition, a sequent has a proof if and only if it has a cut-free proof. (Tiu 2003: also when induction and coinduction are added.)

Theorem: Given a *noetherian* definition and a fixed formula B,

$$\vdash \nabla x \nabla y . B\, x\, y \equiv \nabla y \nabla x . B\, x\, y.$$

Theorem: If we restrict to *Horn definitions* (no implication or negation in the body of the definitions) then

1. $\forall$ and ∇ are interchangeable in definitions,
2. For noetherian definitions and fixed B, $\vdash \nabla x . B\, x \supset \forall x . B\, x$.

The infinite case?

If our logic with *def*$\mathcal{L}$ and *def*$\mathcal{R}$ proves a formula, it is true in *all* fixed points of the definition.

If our definition is *noetherian* then the greatest and least fixed points coincide.

Of course, when there is infinite behaviours, we may be interested only in the least fixed point (inductive) or greatest fixed point (co-inductive).

Alwen Tui and Alberto Momigliano have shown how do *inductive definitions* and *coinductive definitions*.

Tui has put these together with ∇ to define a single, large (meta-logic) **LINC** (lambda, induction, nabla, coinduction). His PhD thesis [March 2004] has numerous examples and numerous meta-theorems, including cut-elimination.

I focus here on finite behavior inorder to focus on bindings.

Example: reasoning with an object-logic

If the formula $\forall u \forall v[q\ \langle u, t_1\rangle\ \langle v, t_2\rangle\ \langle v, t_3\rangle]$ follows from the assumptions

$$\forall x \forall y[q\ x\ x\ y] \qquad \forall x \forall y[q\ x\ y\ x] \qquad \forall x \forall y[q\ y\ x\ x]$$

then the terms t_2 and t_3 are equal.

This seems true independent of the extension of the domain of the quantifiers $\forall u \forall v$. That is, it holds for *internal* reasons instead of *external* reasons.

We would like to prove a meta-level formula like

$$\forall x, y, z[pv\ (\hat{\forall} u \hat{\forall} v[q\ \langle u, x\rangle\ \langle v, y\rangle\ \langle v, z\rangle]) \supset y = z]$$

where $pv\ \cdot$ is defined to encode object-level provability.

Example: encoding π calculus

We write the concrete syntax of π-calculus processes as:

$$P := 0 \mid \tau.P \mid x(y).P \mid \bar{x}y.P \mid (P \mid P) \mid (P + P) \mid (x)P \mid [x = y]P$$

We use three syntactic types: n for names, a for actions, and p for processes. The type n may or may not be inhabited.

We assume three constructors for actions: $\tau : a$ and $\downarrow$ and $\uparrow$ (for input and output actions, resp), both of type $n \rightarrow n \rightarrow a$.

Abstract syntax for processes is the usual. Restriction: $(y)Py$ is coded using a constant $nu : (n \rightarrow p) \rightarrow p$ as $nu(\lambda y.Py)$ or as just nuP. Input prefix $x(y).Py$ is encoded using a constant $in : n \rightarrow (n \rightarrow p) \rightarrow p$ as $in\ x\ (\lambda y.Py)$ or just $in\ x\ P$. Other constructors are done similarly.

π-calculus: one step transitions

One-step transition relation is encoded as two predicates. The type of $\cdot \xrightarrow{\cdot} \cdot$ relates p and a and p. The type of $\cdot \xrightharpoondown{\cdot} \cdot$ relates p and $n \to a$ and $n \to p$.

$P \xrightarrow{A} Q$	free actions, $A : a$ $(\tau, \downarrow xy, \uparrow xy)$
$P \xrightharpoondown{\downarrow x} M$	bound input action, $\downarrow x : n \to a$, $M : n \to p$
$P \xrightharpoondown{\uparrow x} M$	bound output action, $\uparrow x : n \to a$, $M : n \to p$

Consider encoding a few one-step rules as definition clauses.

$$\begin{array}{lrcl}
\mathrm{OUTPUT-ACT:} & \bar{x}y.P \xrightarrow{\uparrow xy} P & \triangleq & \top \\
\mathrm{INPUT-ACT:} & x(y).My \xrightharpoondown{\downarrow x} M & \triangleq & \top \\
\mathrm{MATCH:} & [x = x]P \xrightarrow{\alpha} Q & \triangleq & P \xrightarrow{\alpha} Q \\
\mathrm{RES:} & (x)Px \xrightarrow{\alpha} (x)Qx & \triangleq & \forall x.(Px \xrightarrow{\alpha} Qx)
\end{array}$$

Should that last $\forall$ be a ∇? To know, we must leave Horn clauses.

The process $(y)[x = y]\bar{x}z.0$ cannot make any transition because of the scope of y. Thus the following statement should be provable.

$$\forall x \forall z \forall Q \forall \alpha.[((y)[x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \supset \bot]$$

Given the encoding of restriction using $\forall$, this reduces to proving the sequent

$$\{x, z, Q, \alpha\} : \forall y.([x = y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot$$

No matter what is used to instantiate the $\forall y$, we need to consider the case where x and y are instantiated to the same thing (say, w), and this case leads to the non-provable sequent

$$\{z\} : ([w = w](\bar{w}z.0) \xrightarrow{\bar{w}z} 0) \longrightarrow \bot$$

The universal quantifier was not the correct choice.

Scoping is captured precisely by ∇. Change RES to use ∇.

$$\text{RES}: \quad (x)P \xrightarrow{\alpha} (x)Q \quad \stackrel{\triangle}{=} \quad \nabla x.(P \xrightarrow{\alpha} Q)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{x,z,Q,\alpha\} : w \triangleright ([x=w](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \mathit{def}\mathcal{L}}{\{x,z,Q,\alpha\} : . \triangleright \nabla y.([x=y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \nabla\mathcal{L}}{\{x,z,Q,\alpha\} : . \triangleright ((y)[x=y](\bar{x}z.0) \xrightarrow{\alpha} Q) \longrightarrow \bot}\ \mathit{def}\mathcal{L}}{\{x,z,Q,\alpha\} : \longrightarrow . \triangleright ((y)[x=y](\bar{x}z.0) \xrightarrow{\alpha} Q) \supset \bot}\ \supset\mathcal{R}$$

The success of $\mathit{def}\mathcal{L}$ depends on the unification failure of

$$\lambda w.x = \lambda w.w.$$

π-calculus: encoding (bi)simulation

$$\begin{aligned} \mathit{sim}\ P\ Q \triangleq\ & \forall A \forall P'\ \ [P \xrightarrow{A} P' \supset \exists Q'.Q \xrightarrow{A} Q' \wedge \mathit{sim}\ P'\ Q'] \wedge \\ & \forall X \forall P'\ \ [P \xrightarrow{\downarrow X} P' \supset \exists Q'.Q \xrightarrow{\downarrow X} Q' \wedge \forall w.\mathit{sim}\ (P'w)\ (Q'w)] \wedge \\ & \forall X \forall P'\ \ [P \xrightarrow{\uparrow X} P' \supset \exists Q'.Q \xrightarrow{\uparrow X} Q' \wedge \nabla w.\mathit{sim}\ (P'w)\ (Q'w)] \end{aligned}$$

Note that this definition clause is not Horn and helps to illustrate the differences between $\forall$ and ∇.

This clause defines open (bi)simulation.

If one changes the expression $\forall w.\mathit{sim}\ (P'w)\ (Q'w)$ to

$$\forall w.(\forall y(w = y \vee w \neq y) \supset \mathit{sim}\ (P'w)\ (Q'w))$$

then one is encoding late (bi)simulation.

Conclusions

We have shown a simple extension of intuitionistic logic with generic judgments.

This gives rise to a new quantifier ∇ and a richer sequent with explicit local context.

We have illustrated the use of our logic to organize the syntax of the π-calculus.

The fact that names are an open datatype is illustrated by showing that we need to sometimes include the excluded middle over equality for names.

Future Work

A proof search implementation to do symbolic bisimulation, exploiting unification. Work already started by Alwen Tiu using a unification package from Gopalan Nadathur based on his suspension calculus.

Generalizing the results concerning GSOS and congruence of bisimulation for mobile and for higher-order process calculi. Axelle Ziegler is working on this currently for her “stage de DEA”.

Model theoretic semantics.